Amendments to the Specification:

Please replace the paragraph beginning at page 3, line 11, with the following amended paragraph:

The present invention relates to the efficient implementation of a virtual machine in a computing environment. According to one aspect of the present invention, a method for increasing the performance of a virtual machine includes obtaining a program instruction to be executed by the virtual machine, determining ~~when~~ **whether** the program instruction is a branch instruction, and determining ~~when~~ **whether** a basic block is present in a code cache if the program instruction is a branch instruction. The basic block includes code, and the code cache is associated with the virtual machine. Finally, the method includes executing the code included in the basic block when if the basic block is present. In one embodiment, the program instruction is a bytecode and the code cache is a native code cache.

Please replace the paragraph beginning at page 3, line 29, with the following amended paragraph:

According to another aspect of the present invention, a computing system includes a virtual machine which has a code cache and an interpreter. The interpreter is arranged to obtaining a bytecode and to determine ~~when~~ **whether** the bytecode is a branch bytecode. The interpreter is also arranged to determine ~~when~~ **whether** a basic block is present in the code cache if the bytecode is a branch bytecode. The basic block includes native code, and the interpreter causes the native code to be executed when it is determined that the basic block is present. In one embodiment, the interpreter is further arranged to interpret the bytecode when it is determined that the bytecode is not a branch bytecode.

Please replace the paragraph beginning at page 6, line 17, with the following amended paragraph:

With reference to Figures 1a-e, the execution of a source code statement X:=A+B after the source code statement is compiled into bytecodes will be described in

accordance with an embodiment of the present invention. Instructions 104 associated with the source code statement X:=A+B may be executed by a virtual machine. As will be understood by those skilled in the art, a virtual machine may be implemented as a stack based machine that includes a stack 110 with frames 114a-f which are indexed by a stack pointer 118.

Please replace the paragraph beginning at page 6, line 25, with the following amended paragraph:

Once a first instruction 106a of instructions 104 is interpreted by an interpreter 120, a value of A is pushed onto stack 110 at a location 114a identified by stack pointer 118 which points to a "current" location within stack 110, as shown in Figure 1b. After interpreter 120 executes first instruction 106a, interpreter 120 advances to a second instruction 106b, and dispatches second instruction 106b, as shown in Figure 1c. Like first instruction 106a of Figure 1b, second instruction 106b is a load or push instruction. Hence, the execution of second instruction 106b places a value of B onto stack 110 at a location 114g specified by stack pointer 118.

Please replace the paragraph beginning at page 7, line 2, with the following amended paragraph:

As shown in Figure 1d, once interpreter 120 completes an interpretation cycle associated with second instruction ~~106d~~ 106b, a third instruction 106c may be executed. In the described embodiment, third instruction 106c is an add operation which adds values A and B, and pushes the result of adding values A and B onto location 114a of stack 110. It should be appreciated that once the sum of values A and B is determined, the result of the sum is stored on stack 110 such that values A and B are displaced from stack 110. Finally, after interpreter 120 completes another interpretation cycle, a fourth instruction 106d is executed, and the result stored in location 114a is removed as shown in Figure 1d. Once the result is removed from location 114a, the result is effectively placed into location 114c which is associated with variable X.

Please replace the paragraph beginning at page 9, line 4, with the following amended paragraph:

To facilitate the copying of case blocks 252 substantially directly into native code cache 264, labels may be implemented around case blocks 252 to delineate the beginning and end of a specific case block, e.g., case block 252c. By way of example, a beginning label 270 and an ending label 272 may be inserted before case block 252 and at the end of case block 252, respectively. Ending label 272 is placed before a break instruction which typically terminates each case. The use of labels ~~250, 252~~ 270 and 272 enables code which is to be copied into native code cache 264 to be denoted and, hence, readily identified.

Please replace the paragraph beginning at page 11, line 28, with the following amended paragraph:

As mentioned above with respect to Figure 3, a label table is used in one embodiment to determine whether a particular basic block of native code is cached in a native cache. Figure 4 is a diagrammatic representation of an interpreted program, a label table, and a native cache in accordance with an embodiment of the present invention. When an instruction or a bytecode 406a-b in a program 402 is to be executed, it is determined whether there is a basic block 418a-b associated with bytecode 406. A label table 410 is used to effectively map bytecodes 406 to blocks 418 stored within native code cache 414. If there in no entry in label table 410 for a particular bytecode 406, and it is appropriate for the particular bytecode 406 to have an entry, then space may be allocated within native code cache 414 for a block 418 that corresponds to the particular bytecode 406, and a label for the particular bytecode 406 may be created.